Carnegie Mellon
Software Engineering Institute

DoD Legacy System Migration Guidelines

John Bergey
Dennis Smith
Nelson Weiderman

September 1999

Technical Note
CMU/SEI-99-TN-013

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

DoD Legacy System Migration Guidelines

John Bergey
Dennis Smith
Nelson Weiderman

September 1999

Product Line Practice Initiative

Technical Note
CMU/SEI-99-TN-013

Contents

Abstract

DoD systems contain a substantial amount of legacy software that often needs to be evolved or migrated to new hardware and software platforms. This legacy software is important not only because of the large-scale investment it represents, but also because it defines the baseline for describing future desired capabilities and evolutionary growth. The purpose of this report is to provide a set of DoD legacy system migration guidelines. These guidelines are built on lessons learned from actual legacy system migration efforts and they synthesize previous work on the disciplined evolution of legacy systems.

1 Introduction

DoD systems, like those of other large complex organizations, contain a substantial amount of software. It is important to acquire and evolve software so that it can be managed as an investment that grows in value rather than as a liability whose value depreciates over time.

Legacy systems have been developed over many years at a substantial cost. These legacy systems often represent a patchwork of mainframe, minicomputer, and desktop applications, both centralized and distributed, under dispersed control. They can be fragmented by geography, database incompatibilities, and corporate mergers. As a result, it is difficult to satisfy the need for these systems to represent enterprise assets rather than liabilities. Despite this difficulty, it is important to ensure the vitality of legacy system assets by protecting, managing, evolving, and modernizing them. Thus, the DoD has interests in the processes, methods, and tools available for migrating legacy systems to more desirable target systems, especially to modern architectures and product lines.

A previous report [Bergey 99] described a set of reasons why reengineering projects have failed. Two other works [Bergey 97, Bergey 98] describe a structure and context for exploring disciplined approaches to systems evolution. This technical note develops a set of DoD legacy system migration guidelines based, in part, on the insights of these other works.

As discussed in the next section, some of the current guidelines are more relevant to development/system migration organizations, while others apply equally to acquisition organizations.

2 DoD Legacy System Migration Guidelines

Legacy system migration guidelines can be viewed from the perspective of either an acquisition organization, a development/system migration organization, or a combination of both. Each perspective is important for the DoD.

Because most systems are acquired by contracting with vendors, the DoD has two types of concerns from an acquisition perspective. First, it is important to be a smart acquirer, to understand the technical issues well enough to specify the right system and to focus on relevant issues in selecting and monitoring a contractor. Second, the DoD has an interest in its contractors using state-of-the-practice techniques and processes in order to have a greater likelihood of producing a quality product.

In some cases DoD organizations are continuing to build and migrate prototypes and operational systems. In these cases, it is important to understand the perspective of the development/system migration organization in order to make smart technical choices and to follow a disciplined reengineering process.

Ten basic guidelines are presented below. The relevance of these guidelines for acquisition organizations or development/system migration organizations is highlighted when appropriate. For each guideline, the overall rationale is discussed. Several examples of the application of the general guideline to large-scale migration efforts are also presented.

2.1 Guideline #1: Develop a comprehensive strategy with achievable and measurable milestones for each reengineering project

Most organizations have a long-range, high-level strategy when they embark on a reengineering effort. However, it is important that the long-range strategy addresses the right problem. From either an acquisition or development/system migration perspective, it is important to specify realistic outcomes for a reengineering effort, and to relate these outcomes to specific, incremental milestones that are achievable and measurable.

Clearly, high-level strategic choices have a substantial impact on the success or failure of a reengineering project. Just as architectural decisions have a long-lasting impact on the structure and operation of a system, these early strategic reengineering decisions are difficult to change and have repercussions on the overall reengineering result. Therefore, the crucial first steps need to focus on the right problem, and must be linked to realistic milestones and deliverables.

A comprehensive strategy should carefully test all assumptions and give attention to details. In some cases, problems can occur because the wrong problem is being addressed. In other cases, not all of the components and steps are considered. An example of a flawed strategy is one in which an organization chooses to "replace" rather than "repair" a major subsystem while at the same time abandoning corporate knowledge about the legacy system. Another example of an incomplete strategy occurs when an organization chooses a "big-bang" implementation approach that ignores how to incrementally deploy and transition the system into operational use.

The following examples of guidance should often be part of a comprehensive migration strategy:

- Develop contingency plans for the installation of new systems.
- Maintain parallel operations and separate testing facilities for large, complex systems.
- When integrating software engineering environments, verify that integration between the full range of tools has been proven, and that it is appropriate for the scale of the project.
- Manage a significant changeover to a new system, a new processor, or major upgrades as a project with its own line of accountability and its own budget and resources, rather than as a side project that individuals contribute to in their spare time.

By adopting a high-level structure for decision making early in the reengineering process, a number of predictable problems can be avoided. The inputs driving the reengineering decision analysis should include the enterprise strategy, programmatic issues, economic issues, and technical issues. The strategic issues include the value of the effort, the corporate impact, and the timing. Programmatic issues include resources, priorities, contracting, deliverables, schedule, and risk. Technical issues include feasibility, approach, architecture, tools, and risk. Economic issues include cost, make/buy decisions, and return on investment.

2.2 Guideline #2: When outside systems engineering services are needed, carefully define and monitor their roles

Contractors for systems engineering services can often offer substantial benefits for a number of reasons, such as an understanding of the domain, technical expertise, objectivity, the ability to bring extra personnel to a project quickly, or simply the ability to bring a fresh perspective to a situation. However, if used unwisely, they can also contribute to the failure of reengineering projects. Consequently, their role needs to be carefully defined and monitored.

The DoD and outside systems engineering contractors can have conflicting interests. The former obviously wants to minimize the cost of external resources, while the latter wants to maximize it. It is important for the contracting organization to retain sufficient insight into the work to know if the project is headed for trouble. Systems engineering contractors need to have the right experience and credibility to be of benefit. In addition they need to be given adequate resources and time to complete the job. DoD management should be careful to honestly consider the contractor's recommendations, and should not judge these

recommendations in terms of a preconceived bias. When different contractors are engaged for a project, either in parallel or sequentially, it is important to focus on similarities and differences between their analyses. If different groups identify similar issues or problems, those issues probably merit serious consideration.

The following are examples of specific guidance on the use of contractors:

- Maintain control over the status and direction of a project.
- Be sure that the people actually doing the work are qualified for the task.
- Carefully monitor time and material contracts, especially with regard to
  - analysis of tradeoffs, costs and benefits of requirements
  - understanding whether milestones are meaningful
  - monitoring milestones

2.3 Guideline #3: If new technology is used for a project, provide adequate training in both the technical content and the motivation for change

Although this guideline applies primarily to a system development/migration organization, it has relevance to an acquisition organization. In the case of a breakdown in the technology, the system schedule can be severely impacted. Thus, at a minimum, the contract will need to specify minimum skill levels in the applicable technology. In fact, actual experience with the proposed technologies can be one of the requirements of the past performance volume in the contractor's proposal and can play a part in the technical evaluation criteria. In addition, if the system is to be later maintained by the acquisition organization then its own personnel and support contractors should be fully trained in the relevant technology.

Often, a reengineering effort will take advantage of newer technology than exists in the legacy system. Frequently, the hardware will be changed and updated, and new programming paradigms will be adopted. New vocabularies will be introduced for new ways of doing business.

For example, many new systems will be based on the Internet, the Web, and distributed component technologies. Old systems are often based on a functional style of programming using mainframe computers and radically different file systems. New programming languages, new database paradigms, and new user interfaces are commonly adopted.

It is simply not possible to continue to do business as usual while at the same time bringing the same work force up to speed on the new technologies. Either there must be a conscientious and persistent effort to upgrade the skills of the existing work force, or there must be a replacement of the existing work force, or there must be new workers added to the work force, or some combination of the three.

In planning training, organizations need to consider the technical skills required for adopting new technologies, while also addressing motivational issues. The technical skills can be obtained through courses offered by vendors, outside consulting organizations, or an internal training department. Motivational issues include potential resistance to change, and the organization should present its rationale for the changes, as well as the potential benefits for the employees in using the new skills.

The following are examples of training-related guidance:

- When introducing comprehensive standards, such as the Defense Information Infrastructure Common Operating Environment (DII COE) [DISA 97] or High Level Architecture (HLA) [DMSO 98], be sure that employees understand the full significance and limitations of the change, and that this understanding goes well beyond the ability to use buzzwords.
- When applying new technology with an aging workforce, be sure that the retraining plan is comprehensive, and that it includes additional staff if needed.
- Be prepared to address dysfunctional situations, for example where an organizational culture has grown accustomed to inefficiency, and where greater efficiency is perceived to be a threat to job security.

Thus, training is important, and should encompass both technology and organizational aspects. A development/systems migration organization should answer the following questions:

- Are the training needs of the systems engineers and software engineers identified?
- Is the cost, schedule, and impact of applying the new technology acceptable?
- Is adequate training available?
- Are key members of the project team already well versed in the technology?
- Can they act as mentors to other team members?

2.4 Guideline #4: Establish and maintain configuration management control of the legacy system

Before a system can be managed effectively, a system baseline under configuration management should be in place to aid in disciplined evolution. The system should be well documented and the priority of change requests and their impact on the system should be well understood. In addition, the following items need to be in place: data on the costs of maintaining the system, adequate configuration management, and planning and project management capabilities. If these capabilities are not available, the maintenance effort will become crippled and chaotic, and long-term planning will be problematic.

The heritage of a legacy system can have a large influence on the success or failure of a reengineering project. Many legacy systems are not under adequate control, because the systems are poorly documented and have inadequate historical measurements and inadequate

change control processes. In these cases, it is difficult to understand the current system, to manage it, and to manage changes or plan evolution.

One barometer of whether a system is under control is the way in which change requests are handled. If change requests are prioritized according to their importance and difficulty, rational decisions can be made for new releases. Historical data on similar types of changes, as well as on the ease or difficulty of changes to the affected modules provide an important baseline for enabling changes within schedule and resource constraints.

Another indicator of control is the availability of historical metrics. These include changes that have been made, the costs of those changes, and any problem areas that have occurred. Because every organization is unique, it is important for the data to reflect the unique process and personnel of that organization. For example, data on the initial cost of each component, the size of the component, change history, types of errors, and costs of making changes provide an important part of this baseline.

When such data is not available, it is difficult to make meaningful cost projections for various classes of changes to the system, or to be able to plan on any kind of long-term changes. This results in new releases coming in late, without adequate functionality. It also makes migration efforts impossible to plan.

The following are specific types of guidance for keeping the legacy system under control:

- Carefully track historical data and maintain accurate metrics so that estimates will be accurate, rather than guesses.
- Develop a documented process for managing, tracking and assigning priorities to change requests as a prerequisite for making any type of reengineering effort.

Bergey subdivides the legacy system into three parts: the core system, the operational environment, and the support environments [Bergey 97]. Based on this distinction, he next develops a set of checklists to identify the type of data that is needed for understanding the legacy system and maintaining control over it.

2.5 Guideline #5: There should be a carefully defined and documented process for the elicitation and validation of requirements

It is well known that elicitation and validation of requirements are critical for the success of reengineering efforts. However, far too frequently, system failure is still caused by too little elicitation and validation of requirements for the reengineering effort, as well as by significant flaws in the requirements elicitation and validation process. It is important to have a documented concept of operations for the target system that has the buy-in of key

stakeholders (e.g., external and internal customers, external and internal users, domain experts, and the project team).

Requirements specification is a thorny problem even for "green field" developments (i.e., development from scratch). There are functional requirements and non-functional requirements, user requirements and customer requirements, hardware requirements and software requirements, architecture requirements, maintenance requirements, and logistical requirements. All of these reflect the fact that requirements are not unidimensional. Requirements are not only an expression of the needs of the intended users, but of the many stakeholders who have a vested interest in the system.

For reengineering efforts there are additional problems in eliciting and validating requirements because a requirements baseline for the legacy system frequently does not exist. In the relatively few cases where there may be one, the requirements are typically out of date and do not correspond well to system functionality. Mistakenly assuming that the existing requirements baseline is in good shape, or assuming that there is a small requirements delta when there is actually a large delta, can be disastrous.

The following are examples of guidance to help in requirements elicitation and validation:

- Develop a concept of operations (CONOPS) for the new system to describe the proposed system from a user's operational perspective.
- Include end-to-end operational scenarios of how the new system is to operate from a functional standpoint and fulfill the needs of a diverse set of users.
- Specify the desired system quality attributes (e.g., performance, security, interoperability, and modifiability) that correspond to stakeholder needs.
- Specify operational scenarios for each of the desired system quality attributes and include criteria that can be used to determine if the system satisfies these quality attributes.
- Understand the full implications of a set of requirements or changes because few requirements or changes are "simple" and neatly segmented; rather, they are often intertwined and necessitate significant tradeoffs involving the system quality attributes.
- Be sure that requirements and CONOPS documents are "living" documents, rather than simply documents that are produced to sign off on a milestone.

Today there are many practices aimed at doing a better job of defining requirements. These practices include creating user scenarios, rapid prototyping elements of the system, or developing storyboards to better define the user interface and obtain greater insight into the desired system features and functionality. For example, "use cases" [Jacobsen 92] are now often used to capture requirements in a database and can be used effectively to elicit deltas and to validate requirements. Each of these practices is as appropriate for reengineering as for new development. Some questions to consider in performing requirements include:

- Is there a concept of operations to describe the proposed target system?

- Are the system quality attributes suitably specified?
- Have operational scenarios been developed to describe how the proposed system will operate and how the system quality attributes will be evaluated?
- Have the concept of operations, operational scenarios, and system quality attributes been validated with key stakeholders (e.g., customers, users, system operators) to ensure they accurately reflect their needs?

2.6 Guideline #6: Make software architecture a primary reengineering consideration

A methodical evaluation of the software architectures of the legacy and target systems should be a driving factor in the development of the reengineering technical approach. This evaluation is necessary to determine whether the legacy software architecture is viable at all as a base for further development. It may turn out that the best decision is to throw out the existing system and start from scratch.

Bass defines software architecture as follows:

> "The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them" [Bass 98].

If the existing architecture represents a viable starting point, the reengineering technical approach must be grounded in that architecture. To do otherwise is to start a "green field" development and abandon the previous heritage. Most architectures are by nature long-lived and slow to change. Unless the legacy system architecture is well understood, it becomes very difficult to build a new compatible architecture. If the old architecture is well understood, it becomes possible, for example, to use the existing interfaces to wrap components for use in the new architecture. If the old architecture is well documented, it is possible to use the same types of documentation for the new architecture. Failure to evaluate the existing architecture will lead to inconsistencies between the legacy and target systems. It will also lead to more work and potential problems.

The following are specific areas of architecture guidance:

- Do not upgrade subsystems of a larger system without considering how those subsystems fit into the broader system architecture.
- Do not expect that separately conceived systems can be "federated" until and unless there is an integrating framework architecture to clearly define rules, standards, and protocols for the communication among individual federates.
- Make sure there is a common understanding and representation of the architecture among all the stakeholders.

- Be sure to include software architecture evaluations as formal checkpoints early in the system development phase—when critical design decisions are being made and the architecture is still amenable to change—to minimize risk downstream. This can be done even if the work is being performed under contract.

If the architecture is being acquired, the following guidance should also be considered:

- Include architecture experience as one of the topics to be covered in the past performance volume of the contractor's proposal submission and include it as part of the technical evaluation criteria.
- Conduct an architecture evaluation as part of the source selection process and include the results as part of the technical evaluation criteria.

Unless there is a deep understanding of the core system and its operating environment, the requirements of the system evolution effort cannot be understood. The legacy system has customers, customer sites, and user groups. Interfacing systems, networks, and both internal and external users and usage patterns must be considered. In addition, interoperability considerations, security measures, and logistics need to be evaluated. Much thought needs to be given to formulating the procedural and organizational underpinnings for defining and expressing an architecture. Boehm describes some of the hazards of potential model clashes in architecture descriptions [Boehm 99].

2.7 Guideline #7: There should be a separate and distinct reengineering process

The process by which a legacy system evolves must be carefully defined. The existence of a documented *life cycle process* and corresponding work products are often wrongly viewed as evidence of a sound *reengineering process*. Although work products are a necessary outcome of a reengineering process, there should be a set of tasks and guidance to perform each step, as well as an understanding of how pieces of the whole fit together. In addition, it is necessary to take a broad "reengineering-in-the-large" view that integrates the processes and work products for the entire project. When these processes and products are absent, or are simply hollow exercises, a reengineering project can be severely hampered.

Four critical elements of any reengineering undertaking are the people, the technology, the process, and the resources available. Quality people, with ample resources, employing suitable technologies nevertheless rarely produce a quality product without using a quality process. Although reengineering may be viewed as a relative newcomer on the technical scene, it is no different from any other engineering discipline from the standpoint of being dependent on proven processes. The elements of a software reengineering process closely parallel, but do not duplicate, those of the classic software development process. Reengineering involves a higher degree of "constrained problem solving" because of the fact that the legacy system is the starting point. Typically this entails reverse engineering the

legacy system software to obtain comprehensive program understanding as a precursor to reengineering the system.

The following are specific areas of process guidance:

- Formulate a reengineering process that is unique to your project. Do not use a generic life cycle process.
- Involve the stakeholders in the formulation of the reengineering process.
- Make sure that the metrics of the reengineering process are tied back to the organization's goals and objectives, rather than just generating data that is logged and forgotten.

The crux of this guidance is to adopt a "reengineering-in-the-large" perspective as opposed to considering more narrowly focused aspects such as the reengineering of specific software applications and components. While reengineering the software applications is a critical element, it is a lower-level task that can be defined once the major reengineering considerations are resolved, such as defining the operational system concept, migration strategy, and software architecture for the target system. The goal is to systematically evaluate the major elements that contribute to the reengineering problem and solution space and evaluate how well the organization and project are equipped to perform the reengineering tasks.

If the work is being performed under contract, the acquiring organization should consider a software capability evaluation (SCE) to assess the maturity of the contractor's software development processes. This can provide valuable insight into many of the considerations described in this guideline. The results of the SCE can be factored into the technical evaluation criteria for source selection. An additional requirement could be imposed to specifically describe how the process for a reengineering project differs from that of a conventional software development effort. Previous experience on major reengineering projects could also be a requirement to be addressed under the past-performance volume of the contractor's proposal.

2.8 Guideline #8: Create a team-oriented reengineering plan ... and follow it

A project team must develop a reengineering plan and that plan must then be meticulously carried out. In developing a migration approach for reengineering legacy systems there are many global issues that often need to be resolved by an interdisciplinary team of engineers and domain experts in concert with the software reengineering decision-making. The project team must develop a greater understanding of the legacy system, its mission, the operational environment it is deployed in, and the users of the system and must also have a keen understanding of the organization's goals and objectives for reengineering the system.

It is essential that the documented project plan have the buy-in of key stakeholders (e.g., organizational line managers, the project team, domain experts, and systems and software

practitioners). Major reengineering changes have many steps and involve actions on the parts of all the major stakeholders. Sometimes these plans are not written down and exist only in the minds of some key people, but with the passage of time plans decay.

The following are specific areas of team guidance:

- Appoint and empower a leader and planning group for the planning effort.
- Make sure a complete, written, approved, and funded plan has buy-in from all the major stakeholders.
- Appoint and empower a leader to ensure that the plan is being monitored and updated as necessary with feedback to upper management on a regular basis.

The resolve for executing a plan is provided when the team responsible for executing the plan is given the responsibility for developing the plan. In many cases, the lack of resolve can be traced to a lack of confidence in the plan or a poor plan. A specific part of the plan should be to get the "buy-in" from stakeholders. Once that has been achieved, the plan will roll on its own accord.

Acquisition organizations should specify a reengineering plan as a deliverable instead of a traditional software development plan. Consideration should be given to

- requiring bidding contractors to submit a draft reengineering plan (as part of their proposal) and a plan of action describing how they will fine-tune the technical details and develop a final reengineering plan upon contract award.
- requiring that the draft plan include incremental milestones, interim deliverables, discrete checkpoints for performing an architecture evaluation, a program risk assessment, and a set of metrics to enable the acquiring organization to assess incremental progress.
- evaluating the draft reengineering plan as part of the technical evaluation criteria for source selection.
- making the draft reengineering plan contractually binding if the bidder wins the contract.
- specifying that the detailed, finished plan is to be the first contract deliverable and is to be placed under configuration management control.

2.9 Guideline #9: Management needs to be committed for the long haul

Management must support the plan until it is completed. Management support of the project means careful monitoring and putting things back on track when they stray off track. If management becomes distracted by other projects during the course of a major reengineering effort, it will not know when things go wrong. Of course, management commitment is a generic problem that is common to all large-scale projects, even those outside the domain of software engineering. What makes it particularly important to reengineering is that the consequences of missteps in the early and middle stages can be catastrophic because errors

are so hard to correct when they are found late in the process. Yet management may be prone to be less involved after plans are approved, resources are allocated, and implementation gets underway.

Management must not abrogate its responsibility by throwing the problem "over the fence" or by failing to stay closely involved and adequately supportive of the project. When management is not fully committed to a reengineering effort, the project tends to lose its focus. If management gives up responsibility to lower-level managers or outsiders, they will tend to take the project in different directions than were initially intended. In short, management must be fully focused and fully engaged, and must not become distracted by other high-priority projects.

The following are specific areas of management guidance:

- Do not assign a manager for a reengineering project unless that manager can reasonably be expected to see the project to completion and be responsible for its success.
- Establish a core of personnel that can be counted on for the duration of the project.
- Ensure that the manager does not have so many responsibilities that his or her focus on this particular reengineering project gets diluted.

The realm of management activities includes strategic and business planning, marketing and customer liaison, information technology planning, budgeting and managing resources, organizing and coordinating tasks, overseeing and evaluating projects, and managing infrastructure support. All these activities take long-term commitment and support on the part of a manager. There is no place for half-hearted efforts. There are key work products that need to be produced, key organization processes that must be followed, and infrastructure support that must be built and maintained. Each of these key elements should be communicated through the management reporting chain. If the organization is an acquiring organization, these considerations could be addressed as an integral part of the reengineering plan that is to be submitted as part of the contractor's proposal.

2.10 Guideline #10: Management edicts should not override technical realities

Within the DoD, there are often a large number of stakeholders at different levels, sometimes representing different organizations. On the one hand, this can lead to decisions based on local organizational self-interests or individual agendas; on the other hand, it can lead to decisions based on the wish list of a decision-maker, rather than technical reality. In an acquisition organization, such as the DoD, this guideline primarily applies to the requirements and elicitation and validation phase and the subsequent creation of a robust system specification, request for proposal and contractual statement of work.

Management decisions must be grounded in technical reality. Managers must learn about cost, schedule and capability tradeoffs through their technical advisors. Mandates or edicts

issued by upper management that predetermine the dependent variables without sufficient project team input or concurrence are not conducive to successful completion of a reengineering project. Yet management may be prone to push for early results and early completion often at the expense of careful design analysis.

More often than we would like to admit, project schedules, costs, and deliverables are dictated by top management decisions. Software is a difficult business, especially when one must deal with legacy systems that may have poorly developed components and poor documentation. While top managers do need to make decisions on the allocation of scarce resources, they are often tempted to also determine specific deliverables and timetables. However, detailed planning of schedules and milestones can only be accurately determined through careful study and analysis of the technical parameters of a system, based on an understanding of the system, historical data, and knowledge of the specific skills of the staff. Rather than prescribe these details with little data other than hunches, managers must listen carefully to the technical details and understand the decision-making consequences.

Although the outcome of results based on insufficient analysis is very predictable, the subsequent problems are often mistakenly attributed to the technical decision-making process, the technology that was used, or the reengineering team.

The following are specific areas of tradeoff guidance for managers:

- Be aware of the tradeoffs between schedule, cost, and capability. In any realistic plan improving one of these items will result in a degradation of at least one of the other two.
- Allow at least one free variable among schedule, cost, and deliverables. Management edicts cannot predetermine all three.
- When changes occur to the planned schedule, cost, or function, always consider the impact on the other two.
- Have a clear understanding of the tradeoff between current development costs and future maintenance costs.
- Do not expect that changing technical managers will solve problems that stem from a deficient plan.

Technical tradeoffs have a significant impact on the conduct of any reengineering initiative. Management activities as well as management infrastructure support must accept the realities of these technical tradeoffs and not specify solutions before sufficient analysis has been undertaken. Among the areas that deserve attention are

- an analysis of the costs and benefits of evolving the legacy system
- the feasibility of evolving the legacy system
- an analysis of the independent variables of the solution space
- an analysis of the system engineering and architectural considerations

Unsupported assumptions concerning these important aspects of a reengineering project should be avoided at all cost. As tempting as it may be to make an "educated guess" based on previous experience, technical analysis is always the better course.

Management may want to consider having the reengineering team draft a plan-of-action for conducting an initial analysis to investigate what will be required, evaluate candidate approaches and architectures, and determine the best strategy for reengineering the system. Based on the analysis results, management can finalize an implementation plan and determine the cost, schedule, and required resources with a higher degree of certainty and reduced risk.

3 Summary

DoD organizations have a large stake in legacy systems. In the past, the maintenance, evolution and reengineering of legacy systems have often been managed with far less rigor than the development of new systems. However, because of the magnitude of the investments in legacy systems, as well as their sheer technical complexity, it may be even more important to exercise strong discipline in the migration of legacy systems than in the development of new systems.

Using lessons learned from previous reengineering work, the guidelines presented here can provide a starting point for ensuring that migration projects attain their goals.

References

[Bass 98] Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture in Practice*. SEI Series in Software Engineering. Reading, Ma.: Addison-Wesley, 1998.

[Bergey 97] Bergey, John K.; Northrop, Linda M.; & Smith, Dennis B. *Enterprise Framework for the Disciplined Evolution of Legacy Systems* (CMU/SEI-97-TR-007). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University. Available WWW:
<URL: http://www.sei.cmu.edu/reengineering/pubs/97-TR-007/> (1997).

[Bergey 98] Bergey, John K. *System Evolution Checklists Based on an Enterprise Framework* (white paper). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University. Available WWW:
<URL: http://www.sei.cmu.edu/reengineering/pubs/white-papers/Berg98/> (February 1998).

[Bergey 99] Bergey, John H.; Smith, Dennis B.; Tilley, Scott R; Weiderman, Nelson H.; Woods, Steven G. *Why Reengineering Project Fail.* (CMU/SEI-99-TR-010, ADA 362725). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University. Available WWW:
<URL:
http://www.sei.cmu.edu/publications/documents/99.reports/99tr010/99tr010abstract.html> (1999).

[Boehm 99] Boehm, Barry & Port, Ed. *Escaping the Software Tar Pit: Model Clashes and How to Avoid Them* (Technical Report USC-CSE-98-517). Los Angeles: University of Southern California. Available WWW:
<URL: http://sunset.usc.edu/TechRpts/electronicopy.html> (1999).

[DISA 97] Defense Information Systems Agency. *Defense Information Infrastructure (DII) Common Operating Environment (COE) Baseline Specifications*, Version 3.1. Defense Information Systems Agency. Available WWW: <URL: http://dii-sw.ncr.disa.mil/coe/> (March 29, 1997).

[DMSO 98] Defense Modeling and Simulation Office. *High Level Architecture Technical Specifications*, Version 1.3. Alexandria, Va.: Defense Modeling and Simulation Office. Available WWW: <URL: http://hla.dmso.mil/> (February 1998).

[Jacobsen 92] Jacobsen, Ivar, et al. *Object-Oriented Software Engineering: A Use-Case Driven Approach.* Wokingham, England: Addison-Wesley, 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Final

4. TITLE AND SUBTITLE
DoD Legacy System Migration Guidelines

5. FUNDING NUMBERS

6. AUTHOR(S)
John Bergey, Dennis Smith, Nelson Weiderman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

7. PERFORMING ORGANIZATION REPORT NUMBER
CMU/SEI-99-TN-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12.A DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited, DTIC, NTIS

12.B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

DoD systems contain a substantial amount of legacy software that often needs to evolve or to be migrated to new hardware and software platforms. This legacy software is important not only because of the large scale of the investment that it represents, but also because it represents a baseline for future growth. The purpose of this report is to provide a set of DoD legacy system migration guidelines. These guidelines are built on lessons learned from actual legacy system migration efforts and they synthesize previous work on the disciplined evolution of legacy systems.

14. SUBJECT TERMS	15. NUMBER OF PAGES
enterprise framework, legacy systems, reengineering	30 pp.
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL